

AD-A194 937

ADA COMPILER VALIDATION SUMMARY REPORT: SIEMENS AG

1/1

SIEMENS DS2000 ADA COM. (U)

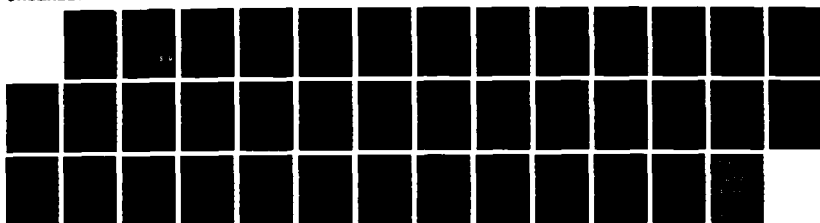
INDUSTRIEANLAGEN-BETRIEBSGESELLSCHAFT M B H OTTOBRUNN

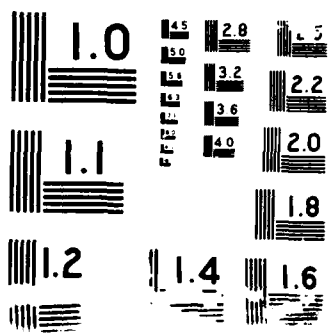
UNCLASSIFIED

(GERMAN ... 26 FEB 87

F/G 12/5

ML





AVF Control Number: AVF-VSR-G09

AD-A194 937

Ada\* COMPILER  
VALIDATION SUMMARY REPORT:  
Siemens AG  
Siemens BS2000 Ada Compiler, Version 1.0  
Siemens 7.570P

Completion of On-Site Testing:  
87-02-26

Prepared By:  
Industrieanlagen-Betriebsgesellschaft mbH, Dept. SZT  
Einsteinstrasse 20  
D-8012 Ottobrunn  
Federal Republic of Germany

Prepared For:  
Ada Joint Program Office  
United States Department of Defense  
Washington, D.C.

DTIC  
ELECTE  
DEC 29 1987  
S H D

\* Ada is a registered trademark of the United States Government (Ada Joint Program Office).

DISTRIBUTION STATEMENT A

Approved for public release;  
Distribution Unlimited

87 12 14 165

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

A194937

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) Ada Compiler Validation Summary Report: Siemens AG Siemens BS2000 Ada Compiler, Version 1.0 Siemens 7.570p		5. TYPE OF REPORT & PERIOD COVERED 26 Feb.'87 to 26 Feb.'88
7. AUTHOR(s) Industr.-Betrieb.		6. PERFORMING ORG. REPORT NUMBER
9. PERFORMING ORGANIZATION AND ADDRESS Industr.-Betrieb., mbH, Dept. SZT Einsteinstrasse 20, D-8012 Ottobrunn Fed. Rep. Germany		8. CONTRACT OR GRANT NUMBER(s)
11. CONTROLLING OFFICE NAME AND ADDRESS Ada Joint Program Office United States Department of Defense Washington, DC 20301-3081ASD/SIOL		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office) Industr.-Beitrieb..		12. REPORT DATE 26 Feb.'87
		13. NUMBER OF PAGES 35
		15. SECURITY CLASS (of this report) UNCLASSIFIED
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE N/A
16. DISTRIBUTION STATEMENT (of this Report)  Approved for public release; distribution unlimited.		
17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20. If different from Report)  UNCLASSIFIED		
18. SUPPLEMENTARY NOTES		
19. KEYWORDS (Continue on reverse side if necessary and identify by block number)  Ada Programming language, Ada Compiler Validation Summary Report, Ada Compiler Validation Capability, ACVC, Validation Testing, Ada Validation Office, AVO, Ada Validation Facility, AVF, ANSI/MIL-STD- 1815A, Ada Joint Program Office, AJPO		
20. ABSTRACT (Continue on reverse side if necessary and identify by block number)  See Attached.		

DD FORM

1473

EDITION OF 1 NOV 65 IS OBSOLETE

1 JAN 73

S/N 0102-LF-014-6601

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

AVF Control Number: AVF-VSR-009

Ada\* COMPILER  
VALIDATION SUMMARY REPORT:  
Siemens AG  
Siemens PS2000 Ada Compiler, Version 1.0  
Siemens 7.570P

Completion of On-Site Testing:  
87-02-26

Prepared By:  
Industrieanlagen-Betriebsgesellschaft mbH, Dept. SZT  
Einsteinstrasse 20  
D-8012 Ottobrunn  
Federal Republic of Germany



Prepared For:  
Ada Joint Program Office  
United States Department of Defense  
Washington, D.C.

Accession For	
NTIS GRA&I	<input checked="checked" type="checkbox"/>
DTIC TAB	<input type="checkbox"/>
Unannounced	<input type="checkbox"/>
Justification	
By	
Distribution/	
Availability Codes	
Avail and/or	
Dist	Special
A-1	

\* Ada is a registered trademark of the United States Government (Ada Joint Program Office).

# Ada\* Compiler Validation Summary Report:

Compiler Name: Siemens BS2000 Ada Compiler, Version 1.0

Host:

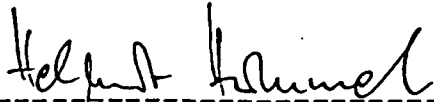
Siemens 7.570P under  
BS2000,  
Version 7.6

Target:

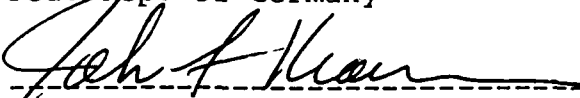
Siemens 7.570P under  
BS2000,  
Version 7.6

Testing Completed 87-02-26 Using ACVC 1.8

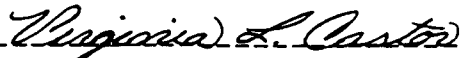
This report has been reviewed and is approved.



IABG m.b.H., Dept SZT  
Dr. H. Hummel  
Einsteinstrasse  
D 8012 Ottobrunn  
Fed. Rep. of Germany



Ada Validation Organization  
Dr. John F. Kramer  
Institute for Defense Analyses  
Alexandria VA



Ada Joint Program Office  
Virginia L. Castor  
Director  
Department of Defense  
Washington DC

\* Ada is a registered trademark of the United States Government (Ada Joint Program Office).

## EXECUTIVE SUMMARY

This Validation Summary Report (VSR) summarizes the results and conclusions of validation testing performed on the Siemens BS2000 Ada Compiler, Version 1.0, using Version 1.8 of the Ada Compiler Validation Capability (ACVC). The Siemens BS2000 Ada Compiler is hosted on a Siemens 7.570P operating under BS2000, Version 7.6. Programs processed by this compiler may be executed on a Siemens 7.570P operating under BS2000, Version 7.6.

On-site testing was performed 87-02-25 through 87-02-26 at Siemens AG in D-8000 München-Neuperlach, under the direction of the Industrieanlagen-Betriebsgesellschaft mbH, Dept. SZT (AVF), according to Ada Validation Organization (AVO) policies and procedures. The AVF identified 2210 of the 2399 tests in ACVC Version 1.8 to be processed during on-site testing of the compiler. The 19 tests withdrawn at the time of validation testing, as well as the 170 executable tests that make use of floating-point precision exceeding that supported by the implementation, were not processed. After the 2210 tests were processed, results for Class A, C, D, or E tests were examined for correct execution. Compilation listings for Class B tests were analyzed for correct diagnosis of syntax and semantic errors. Compilation and link results of Class L tests were analyzed for correct detection of errors. There were 37 of the processed tests determined to be inapplicable. The remaining 2173 tests were passed.

The results of validation are summarized in the following table:

RESULT	CHAPTER													TOTAL
	2	3	4	5	6	7	8	9	10	11	12	14		
Passed	102	250	334	243	161	97	134	262	124	32	218	216	2173	
Failed	0	0	0	0	0	0	0	0	0	0	0	0	0	
Inapplicable	14	75	86	4	0	0	5	0	6	0	0	17	207	
Withdrawn	0	5	5	0	0	1	1	2	4	0	1	0	19	
TOTAL	116	330	425	247	161	98	140	264	134	32	219	233	2399	

The AVF concludes that these results demonstrate acceptable conformity to ANSI/MIL-STD-1815A Ada.

---

\* Ada is a registered trademark of the United States Government (Ada Joint Program Office).

## TABLE OF CONTENTS

		Page
CHAPTER 1	INTRODUCTION	5
1.1	PURPOSE OF THIS VALIDATION SUMMARY REPORT	6
1.2	USE OF THIS VALIDATION SUMMARY REPORT . .	6
1.3	REFERENCES . . . . .	7
1.4	DEFINITION OF TERMS . . . . .	7
1.5	ACVC TEST CLASSES . . . . .	8
CHAPTER 2	CONFIGURATION INFORMATION	11
2.1	CONFIGURATION TESTED . . . . .	11
2.2	IMPLEMENTATION CHARACTERISTICS . . . . .	11
CHAPTER 3	TEST INFORMATION	16
3.1	TEST RESULTS . . . . .	16
3.2	SUMMARY OF TEST RESULTS BY CLASS . . . . .	16
3.3	SUMMARY OF TEST RESULTS BY CHAPTER . . . . .	16
3.4	WITHDRAWN TESTS . . . . .	17
3.5	INAPPLICABLE TESTS . . . . .	17
3.6	SPLIT TESTS . . . . .	18
3.7	ADDITIONAL TESTING INFORMATION . . . . .	19
3.7.1	Prevalidation . . . . .	19
3.7.2	Test Method . . . . .	19
3.7.3	Test Site . . . . .	20
APPENDIX A	COMPLIANCE STATEMENT	21
APPENDIX B	APPENDIX F OF THE Ada STANDARD	23
APPENDIX C	TEST PARAMETERS	31
APPENDIX D	WITHDRAWN TESTS	35

## CHAPTER 1

### INTRODUCTION

This Validation Summary Report (VSR) describes the extent to which a specific Ada compiler conforms to the Ada Standard, ANSI/MIL-STD-1815A. This report explains all technical terms used within it and thoroughly reports the results of testing this compiler using the Ada Compiler Validation Capability (ACVC). An Ada compiler must be implemented according to the Ada Standard, and any implementation-dependent features must conform to the requirements of the Ada Standard. The Ada Standard must be implemented in its entirety, and nothing can be implemented that is not in the Standard.

Even though all validated Ada compilers conform to the Ada Standard, it must be understood that some differences do exist between implementations. The Ada Standard permits some implementation dependencies -- for example, the maximum length of identifiers or the maximum values of integer types. Other differences between compilers result from characteristics of particular operating systems, hardware, or implementation strategies. All of the dependencies observed during the process of testing this compiler are given in this report.

The information in this report is derived from the test results produced during validation testing. The validation process includes submitting a suite of standardized tests, the ACVC, as inputs to an Ada compiler and evaluating the results. The purpose of validating is to ensure conformity of the compiler to the Ada Standard by testing that the compiler properly implements legal language constructs and that it identifies and rejects illegal language constructs. The testing also identifies behavior that is implementation dependent but permitted by the Ada Standard. Six classes of tests are used. These tests are designed to perform checks at compile time, at link time, and during execution.

### 1.1 PURPOSE OF THIS VALIDATION SUMMARY REPORT

This VSR documents the results of the validation testing performed on an Ada compiler. Testing was carried out for the following purposes:

- . To attempt to identify any language constructs supported by the compiler that do not conform to the Ada Standard
- . To attempt to identify any unsupported language constructs required by the Ada Standard
- . To determine that the implementation-dependent behavior is allowed by the Ada Standard

Testing of this compiler was conducted by the AVF according to policies and procedures established by the Ada Validation Organization (AVO). On-site testing was conducted from 87-02-25 through 87-02-26 at Siemens AG in D-8000 München-Neuperlach.

### 1.2 USE OF THIS VALIDATION SUMMARY REPORT

Consistent with the national laws of the originating country, the AVO may make full and free public disclosure of this report. In the United States, this is provided in accordance with the "Freedom of Information Act" (5 U.S.C. /552). The results of this validation apply only to the computers, operating systems, and compiler versions identified in this report.

The organizations represented on the signature page of this report do not represent or warrant that all statements set forth in this report are accurate and complete, or that the subject compiler has no nonconformities to the Ada Standard other than those presented. Copies of this report are available to the public from:

Ada Information Clearinghouse  
Ada Joint Program Office  
OUSDRE  
The Pentagon, Rm 3D-139 (Fern Street)  
Washington DC 20301-3081

or from:

Industrieanlagen-Betriebsgesellschaft mbH, Dept. SZT  
Einsteinstrasse 20  
D-8012 Ottobrunn  
Federal Republic of Germany

Questions regarding this report or the validation test results should be directed to the AVF listed above or to:

Ada Validation Organization  
Institute for Defense Analyses  
1801 North Beauregard Street  
Alexandria VA 22311

### 1.3 REFERENCES

1. Reference Manual for the Ada Programming Language, ANSI/MIL-STD-1815A, FEB 1983.
2. Ada Validation Organization: Policies and Procedures, MITRE Corporation, JUN 1982, PB 83-110601.
3. Ada Compiler Validation Capability Implementers' Guide, SofTech, Inc., DEC 1984.

### 1.4 DEFINITION OF TERMS

ACVC	The Ada Compiler Validation Capability. A set of programs that evaluates the conformity of a compiler to the Ada language specification, ANSI/MIL-STD-1815A.
Ada Standard	ANSI/MIL-STD-1815A, February 1983.
Applicant	The agency requesting validation.
AVF	The Ada Validation Facility. In the context of this report, the AVF is responsible for conducting compiler validations according to established policies and procedures.
AVO	The Ada Validation Organization. In the context of this report, the AVO is responsible for setting procedures for compiler validations.
Compiler	A processor for the Ada language. In the context of this report, a compiler is any language processor, including cross-compilers, translators, and interpreters.
Failed test	A test for which the compiler generates a result that demonstrates nonconformity to the Ada Standard.

Host	The computer on which the compiler resides.
Inapplicable test	A test that uses features of the language that a compiler is not required to support or may legitimately support in a way other than the one expected by the test.
Passed test	A test for which a compiler generates the expected result.
Target	The computer for which a compiler generates code.
Test	A program that checks a compiler's conformity regarding a particular feature or features to the Ada Standard. In the context of this report, the term is used to designate a single test, which may comprise one or more files.
Withdrawn test	A test found to be incorrect and not used to check conformity to the Ada language specification. A test may be incorrect because it has an invalid test objective, fails to meet its test objective, or contains illegal or erroneous use of the language.

### 1.5 ACVC TEST CLASSES

Conformity to the Ada Standard is measured using the ACVC. The ACVC contains both legal and illegal Ada programs structured into six test classes: A, B, C, D, E, and L. The first letter of a test name identifies the class to which it belongs. Class A, C, D, and E tests are executable, and special program units are used to report their results during execution. Class B tests are expected to produce compilation errors. Class L tests are expected to produce link errors.

Class A tests check that legal Ada programs can be successfully compiled and executed. However, no checks are performed during execution to see if the test objective has been met. For example, a Class A test checks that reserved words of another language (other than those already reserved in the Ada language) are not treated as reserved words by an Ada compiler. A Class A test is passed if no errors are detected at compile time and the program executes to produce a PASSED message.

Class B tests check that a compiler detects illegal language usage. Class B tests are not executable. Each test in this class is compiled and the resulting compilation listing is examined to verify that every syntax or semantic error in the test is detected. A Class B test is passed if every

illegal construct that it contains is detected by the compiler.

Class C tests check that legal Ada programs can be correctly compiled and executed. Each Class C test is self-checking and produces a PASSED, FAILED, or NOT APPLICABLE message indicating the result when it is executed.

Class D tests check the compilation and execution capacities of a compiler. Since there are no capacity requirements placed on a compiler by the Ada Standard for some parameters -- for example, the number of identifiers permitted in a compilation or the number of units in a library -- a compiler may refuse to compile a Class D test and still be a conforming compiler. Therefore, if a Class D test fails to compile because the capacity of the compiler is exceeded, the test is classified as inapplicable. If a Class D test compiles successfully, it is self-checking and produces a PASSED or FAILED message during execution.

Each Class E test is self-checking and produces a NOT APPLICABLE, PASSED, or FAILED message when it is compiled and executed. However, the Ada Standard permits an implementation to reject programs containing some features addressed by Class E tests during compilation. Therefore, a Class E test is passed by a compiler if it is compiled successfully and executes to produce a PASSED message, or if it is rejected by the compiler for an allowable reason.

Class L tests check that incomplete or illegal Ada programs involving multiple, separately compiled units are detected and not allowed to execute. Class L tests are compiled separately and execution is attempted. A Class L test passes if it is rejected at link time -- that is, an attempt to execute the main program must generate an error message before any declarations in the main program or any units referenced by the main program are elaborated.

Two library units, the package REPORT and the procedure CHECK\_FILE, support the self-checking features of the executable tests. The package REPORT provides the mechanism by which executable tests report PASSED, FAILED, or NOT APPLICABLE results. It also provides a set of identity functions used to defeat some compiler optimizations allowed by the Ada Standard that would circumvent a test objective. The procedure CHECK\_FILE is used to check the contents of text files written by some of the Class C tests for chapter 14 of the Ada Standard. The operation of these units is checked by a set of executable tests. These tests produce messages that are examined to verify that the units are operating correctly. If these units are not operating correctly, then the validation is not attempted.

The text of the tests in the ACVC follow conventions that

are intended to ensure that the tests are reasonably portable without modification. For example, the tests make use of only the basic set of 55 characters, contain lines with a maximum length of 72 characters, use small numeric values, and place features that may not be supported by all implementations in separate tests. However, some tests contain values that require the test to be customized according to implementation-specific values -- for example, an illegal file name. A list of the values used for this validation is provided in Appendix C.

A compiler must correctly process each of the tests in the suite and demonstrate conformity to the Ada Standard by either meeting the pass criteria given for the test or by showing that the test is inapplicable to the implementation. Any test that was determined to contain an illegal language construct or an erroneous language construct is withdrawn from the ACVC and, therefore, is not used in testing a compiler. The tests withdrawn at the time of validation are given in Appendix D.

## CHAPTER 2

### CONFIGURATION INFORMATION

#### 2.1 CONFIGURATION TESTED

The candidate compilation system for this validation was tested under the following configuration:

Compiler: Siemens BS2000 Ada Compiler, Version 1.0

ACVC Version: 1.8

Certificate Expiration Date: 88-05-04

Host Computer:

Machine: Siemens 7.570P

Operating System: BS2000  
Version 7.6

Memory Size: 7 MB virtual/32 MB real

Target Computer:

Machine: Siemens 7.570P

Operating System: BS2000  
Version 7.6

Memory Size: 7 MB virtual/32 MB real

The disk system of the host and target computer consisted of 24 Siemens disks D3475 with 757 MB each and 3 Siemens disks D3468 with 267 MB each.

#### 2.2 IMPLEMENTATION CHARACTERISTICS

One of the purposes of validating compilers is to determine the behavior of a compiler in those areas of the Ada Standard that permit implementations to differ. Class D and E tests specifically check for such implementation differences. However, tests in other classes also characterize an implementation. This compiler is characterized by the following interpretations of the Ada Standard:

- . Capacities.

The compiler correctly processes tests containing loop statements nested to 65 levels, block statements nested to 65 levels, and recursive procedures separately compiled as subunits nested to 17 levels. It correctly processes a compilation containing 723 variables in the same declarative part. (See tests D55A03A..H (8 tests), D56001B, D64005E..G (3 tests), and D29002K.)

- . Universal integer calculations.

An implementation is allowed to reject universal integer calculations having values that exceed `SYSTEM.MAX_INT`. This implementation does not reject such calculations and processes them correctly. (See tests D4A002A, D4A002B, D4A004A, and D4A004B.)

- . Predefined types.

This implementation supports the additional predefined type `SHORT_INTEGER` in the package `STANDARD`. (See tests B86001C and B86001D.)

- . Based literals.

An implementation is allowed to reject a based literal with a value exceeding `SYSTEM.MAX_INT` during compilation, or it may raise `NUMERIC_ERROR` or `CONSTRAINT_ERROR` during execution. This implementation raises `CONSTRAINT_ERROR` during execution. (See test E24101A.)

- . Array types.

An implementation is allowed to raise `NUMERIC_ERROR` or `CONSTRAINT_ERROR` for an array having a '`LENGTH`' that exceeds `STANDARD.INTEGER'LAST` and/or `SYSTEM.MAX_INT`.

A packed `BOOLEAN` array having a '`LENGTH`' exceeding `INTEGER'LAST` raises `NUMERIC_ERROR` when the array type is declared. (See test C52103X.)

A packed two-dimensional `BOOLEAN` array with more than `INTEGER'LAST` components raises `NUMERIC_ERROR` when the array type is declared. (See test C52104Y.)

A null array with one dimension of length greater than `INTEGER'LAST` may raise `NUMERIC_ERROR` or `CONSTRAINT_ERROR` either when declared or assigned. Alternatively, an implementation may accept the declaration. However,

lengths must match in array slice assignments. This implementation raises NUMERIC ERROR when the array type is declared. (See test E52103Y.)

In assigning one-dimensional array types, the expression appears to be evaluated in its entirety before CONSTRAINT ERROR is raised when checking whether the expression's subtype is compatible with the target's subtype. In assigning two-dimensional array types, the expression does not appear to be evaluated in its entirety before CONSTRAINT ERROR is raised when checking whether the expression's subtype is compatible with the target's subtype. (See test C52013A.)

- . Discriminated types.

During compilation, an implementation is allowed to either accept or reject an incomplete type with discriminants that is used in an access type definition with a compatible discriminant constraint. This implementation accepts such subtype indications. (See test E38104A.)

In assigning record types with discriminants, the expression appears to be evaluated in its entirety before CONSTRAINT ERROR is raised when checking whether the expression's subtype is compatible with the target's subtype. (See test C52013A.)

- . Aggregates.

In the evaluation of a multi-dimensional aggregate, all choices appear to be evaluated before checking against the index type. (See tests C43207A and C43207B.)

In the evaluation of an aggregate containing subaggregates, all choices are not evaluated before being checked for identical bounds. (See test E43212B.) All choices are evaluated before CONSTRAINT ERROR is raised if a bound in a nonnull range of a nonnull aggregate does not belong to an index subtype. (See test E43211B.)

- . Functions.

An implementation may allow the declaration of a parameterless function and an enumeration literal having the same profile in the same immediate scope, or it may reject the function declaration. If it accepts the function declarations, the use of the enumeration literal's identifier denotes the function. This implementation rejects the declarations. (See test E66001D.)

- Representation clauses.

The Ada Standard does not require an implementation to support representation clauses. If a representation clause is not supported, then the implementation must reject it. While the operation of representation clauses is not checked by Version 1.8 of the ACVC, they are used in testing other language features. This implementation rejects 'SIZE and 'STORAGE\_SIZE for tasks, 'STORAGE\_SIZE for collections, and 'SMALL clauses. Enumeration representation clauses appear not to be supported. (See tests C55B16A, C87B62A, C87B62B, C87B62C, and BC1002A.)

- Pragmas.

The pragma INLINE is not supported for procedures. The pragma INLINE is not supported for functions. (See tests CA3004E and CA3004F.)

- Input/output.

The package SEQUENTIAL\_IO cannot be instantiated with unconstrained array types and record types with discriminants. The package DIRECT\_IO cannot be instantiated with unconstrained array types and record types with discriminants without defaults. (See tests AE2101C, AE2101H, CE2201D, CE2201E, and CE2401D.)

An existing text file can be opened in OUT\_FILE mode, can be created in OUT\_FILE mode, and can be created in IN\_FILE mode. (See test EE3102C.)

More than one internal file can be associated with each external file for text I/O for reading only. (See tests CE3111A..E (5 tests).)

More than one internal file can be associated with each external file for sequential I/O for reading only. (See tests CE2107A..F (6 tests).)

More than one internal file can be associated with each external file for direct I/O for reading only. (See tests CE2107A..F (6 tests).)

Temporary sequential files are given a name. Temporary direct files are given a name. Temporary files given names are deleted when they are closed. (See tests CE2108A and CE2108C.)

. Generics.

Generic subprogram declarations and bodies can be compiled in separate compilations. (See test CA2009F.) Generic package declarations and bodies can be compiled in separate compilations. (See tests CA2009C and BC3205D.)

### CHAPTER 3

#### TEST INFORMATION

#### 3.1 TEST RESULTS

Version 1.8 of the ACVC contains 2399 tests. When validation testing of Siemens BS2000 Ada Compiler was performed, 19 tests had been withdrawn. The remaining 2380 tests were potentially applicable to this validation. The AVF determined that 207 tests were inapplicable to this implementation, and that the 2173 applicable tests were passed by the implementation. There were no failed tests.

The AVF concludes that the testing results demonstrate acceptable conformity to the Ada Standard.

#### 3.2 SUMMARY OF TEST RESULTS BY CLASS

RESULT	TEST CLASS						TOTAL
	A	B	C	D	E	L	
Passed	67	864	1170	17	11	44	2173
Failed	0	0	0	0	0	0	0
Inapplicable	2	3	198	0	2	2	207
Withdrawn	0	7	12	0	0	0	19
TOTAL	69	874	1380	17	13	46	2399

#### 3.3 SUMMARY OF TEST RESULTS BY CHAPTER

RESULT	CHAPTER												TOTAL
	2	3	4	5	6	7	8	9	10	11	12	14	
Passed	102	250	334	243	161	97	134	262	124	32	218	216	2173
Failed	0	0	0	0	0	0	0	0	0	0	0	0	0
Inapplicable	14	75	86	4	0	0	5	0	6	0	0	17	207
Withdrawn	0	5	5	0	0	1	1	2	4	0	1	0	19
TOTAL	116	330	425	247	161	98	140	264	134	32	219	233	2399

### 3.4 WITHDRAWN TESTS

The following 19 tests were withdrawn from ACVC Version 1.8 at the time of this validation:

C32114A	C41404A	B74101B
B33203C	B45116A	C87B50A
C34018A	C43008A	C92005A
C35904A	B49006A	C940ACA
B37401A	B4A010C	CA3005A..D (4 tests)
		BC3204C

See Appendix D for the reason that each of these tests was withdrawn.

### 3.5 INAPPLICABLE TESTS

Some tests do not apply to all compilers because they make use of features that a compiler is not required by the Ada Standard to support. Others may depend on the result of another test that is either inapplicable or withdrawn. For this validation attempt, 207 tests were inapplicable for the reasons indicated:

- . C34001E, B52004D, B55B09C, and C55B07A use LONG\_INTEGER which is not supported by this compiler.
- . C34001F and C35702A use SHORT\_FLOAT which is not supported by this compiler.
- . C34001G and C35702B use LONG\_FLOAT which is not supported by this compiler.
- . C55B16A makes use of an enumeration representation clause containing noncontiguous values which is not supported by this compiler.
- . B86001DT requires a predefined numeric type other than those defined by the Ada language in package STANDARD. There is no such type for this implementation.
- . C86001F redefines package SYSTEM, but TEXT\_IO is made obsolete by this new definition in this implementation and the test cannot be executed since the package REPORT is dependent on the package TEXT\_IO.
- . C87B62A..C (3 tests) use length clauses which are not supported by this compiler. The length clause is rejected during compilation.

- . CA3004E, EA3004C, and LA3004A use INLINE pragma for procedures which is not supported by this compiler.
- . CA3004F, EA3004D, and LA3004B use INLINE pragma for functions which is not supported by this compiler.
- . AE2101C, CE2201D, and CE2201E use an instantiation of package SEQUENTIAL\_IO with unconstrained array types which is not supported by this compiler.
- . AE2101H and CE2401D use an instantiation of package DIRECT\_IO with unconstrained array types which is not supported by this compiler.
- . CE2107B..E (4 tests), CE2110B, CE2111D, CE2111H, CE3111B..E (4 tests), and CE3114B are inapplicable because multiple internal files cannot be associated with the same external file. The proper exception is raised when multiple access is attempted.
- . The following 170 tests require a floating-point accuracy that exceeds the maximum of 15 supported by the implementation:

C24113L..Y (14 tests)  
C35705L..Y (14 tests)  
C35706L..Y (14 tests)  
C35707L..Y (14 tests)  
C35708L..Y (14 tests)  
C35802L..Y (14 tests)  
C45241L..Y (14 tests)  
C45321L..Y (14 tests)  
C45421L..Y (14 tests)  
C45424L..Y (14 tests)  
C45521L..Z (15 tests)  
C45621L..Z (15 tests)

### 3.6 SPLIT TESTS

If one or more errors do not appear to have been detected in a Class B test because of compiler error recovery, then the test is split into a set of smaller tests that contain the undetected errors. These splits are then compiled and examined. The splitting process continues until all errors are detected by the compiler or until there is exactly one error per split. Any Class A, Class C, or Class E test that cannot be compiled and executed because of its size is split into a set of smaller subtests that can be processed.

Splits were required for 3 Class B tests.

B22003A-S1	B67001C-S5
B67001C-S1	B67001C-S6
B67001C-S2	B67001C-S7
B67001C-S3	B95032A-S1
B67001C-S4	B95032A-S2

### 3.7 ADDITIONAL TESTING INFORMATION

#### 3.7.1 Prevalidation

Prior to validation, a set of test results for ACVC Version 1.8 produced by the Siemens BS2000 Ada Compiler was submitted to the AVF by the applicant for review. Analysis of these results demonstrated that the compiler successfully passed all applicable tests, and the compiler exhibited the expected behavior on all inapplicable tests.

#### 3.7.2 Test Method

Testing of the Siemens BS2000 Ada Compiler using ACVC Version 1.8 was conducted on-site by a validation team from the AVF. The configuration consisted of a Siemens 7.570P operating under BS2000, Version 7.6 with the target being the same machine.

The body of the package REPORT was modified so that a time stamp for each executed test will be added.

A magnetic tape containing all tests except for withdrawn tests and tests requiring unsupported floating-point precisions was taken on-site by the validation team for processing. Tests that make use of implementation-specific values were customized before being written to the magnetic tape. Tests requiring splits during the prevalidation testing were included in their split form on the magnetic tape except for B95032A which was checked by the validation team.

The contents of the magnetic tape were loaded directly onto the host computer using the BS2000 utility SYSUPD.

After the test files were loaded to disk, the full set of tests was compiled on the Siemens 7.570P, and all executable tests were linked and executed on the target (the same machine).

The compiler was tested using command scripts provided by Siemens AG and reviewed by the validation team.

All tests were run using these command scripts. The tests CE3201A, CE3203A, CE3301B, CE3402B, CE3405B, CE3409F, CE3410F, and CE3412C whose self-documenting special messages to STANDARD OUTPUT require a manual check were also run in batch mode. The listings produced of these eight tests were printed on paper by a specific system command procedure.

The following options were in effect for testing:

<u>Option</u>	<u>Effect</u>
70	Suppress end messages of the compiler
412	Check if instantiations are necessary at link time
566	Suppress listing by Ada linker of all linked Ada compilation units

Tests were compiled, linked, and executed (as appropriate) using a single host computer. In order to minimize disk access and thereby enhance throughput, exact copies of the compiler resided on three disk units. Up to eleven batch jobs ran simultaneously each using one of the three Ada compilers. Test output, compilation listings, and job logs were captured on magnetic tape and archived at the AVF. The listings examined on-site by the validation team were also archived.

### 3.7.3 Test Site

The validation team arrived at Siemens AG in D-8000 München-Neuperlach on 87-02-25 and departed after testing was completed on 87-02-26.

APPENDIX A

DECLARATION OF CONFORMANCE

Compiler Implementer: Siemens AG, München  
Ada Validation Facility: IABG m.b.H., Ottobrunn  
Ada Compiler Validation Capability (ACVC) Version: 1.8

Base Configuration

Base Compiler Name: Siemens BS2000 Ada Compiler Version: 1.0  
Host Architecture ISA Siemens 7.570P OS&VER No: BS2000 / V7.6  
Target Architecture ISA: Siemens 7.570P OS&VER No: BS2000 / V7.6

Derived Compiler Registration

Derived Compiler Name: Siemens BS2000 Ada Compiler Version: 1.0  
(same as above)

Host Architecture ISA: 7.531, 7.536, 7.541, 7.551,  
7.530, 7.550, 7.560,  
7.561, 7.571, 7.550,  
7.560, 7.570, 7.580, 7.590,  
7.700

OS&VER No: BS2000 / V7.5, V7.6, V8.0, V8.5, V9.0

Target Architecture ISA: same as host  
OS&VER No: same as host

The above-mentioned compiler executes properly without any modifications on each of these host architectures in combination with each of the BS2000-versions listed above.

Implementer's Declaration

I, the undersigned, representing Siemens A.G. have implemented no deliberate extensions to the Ada Language Standard ANSI/MIL-STD-1815A in the compiler listed in this declaration. I declare that Siemens A.G. is the owner of record of the Ada language compiler listed above and, as such, is responsible for maintaining said compiler in conformance to ANSI/MIL-STD-1815A. All certificates and registrations for Ada language compilers listed in this declaration shall be made only in the owner's corporate name.

Siemens Aktiengesellschaft

S.S. 1987 e.V. 

**Owner's Declaration**

I, the undersigned, representing Siemens A.G. take full responsibility for implementation and maintenance of the Ada compiler listed above, and agree to the public disclosure of the final Validation Summary Report. I further agree to continue to comply with the Ada trademark policy, as defined by the Ada Joint Program Office. I declare that all of the Ada language compilers listed, and their host/target performance are in compliance with the Ada Languages Standard ANSI/MIL-STD-1815A. I have reviewed the Validation Summary Report for the compiler and concur with the contents.

Siemens Aktiengesellschaft

8.5. 1987 c. V. *W. H. H. H. H. H.*

## APPENDIX B

### APPENDIX F OF THE Ada STANDARD

The only allowed implementation dependencies correspond to implementation-dependent pragmas, to certain machine-dependent conventions as mentioned in chapter 13 of MIL-STD-1815A, and to certain allowed restrictions on representation clauses. The implementation-dependent characteristics of the Siemens BS2000 Ada Compiler, Version 1.0, are described in the following sections which discuss topics in Appendix F of the Ada Language Reference Manual (ANSI/MIL-STD-1815A). Implementation-specific portions of the package STANDARD are as follows:

package STANDARD is

...

type INTEGER is range -2147\_483\_647 .. 2\_147\_483\_647;

type SHORT\_INTEGER is range -32768 .. 32767;

type FLOAT is digits 15 range -2#1.0#E212 .. 2#1.0#E212;

type DURATION is delta 2#1.0#E-14 range -131071.0 .. 131071.0;

-- DURATION'SMALL = 2#1.0#E-14.

...

end STANDARD;

#### Implementation-dependent Characteristics

This appendix summarizes all implementation-dependent characteristics of the Siemens BS2000 Ada Compiler. It describes:

- (1) The form, allowed places, and effect of every implementation-dependent pragma.
- (2) The name and the type of every implementation-dependent attribute.
- (3) The specification of the package SYSTEM.
- (4) The list of all restrictions on representation clauses.
- (5) The conventions used for any implementation-generated name denoting implementation-dependent components.
- (6) The interpretation of expressions that appear in address clauses, including those for interrupts.

- (7) Any restriction on unchecked conversions.
- (8) Any implementation-dependent characteristics of the input-output packages.

### 1. Implementation-dependent Pragmas

There are no implementation-dependent pragmas.

The only language name accepted by pragma INTERFACE is ASSEMBLER. The only priority accepted by pragma PRIORITY is represented by an expression of the static value 0 (cf. the definition of the subtype PRIORITY in package SYSTEM).

### 2. Implementation-dependent Attributes

There are no implementation-dependent attributes.

### 3. Specification of the Package SYSTEM

```
package SYSTEM is
  type ADDRESS is new INTEGER;
  type NAME      is (BS2000);
  SYSTEM_NAME    : constant NAME := BS2000;
  STORAGE_UNIT   : constant := 8;
  MEMORY_SIZE    : constant := 4000000;
  -- System-Dependent Named Numbers:
  MIN_INT        : constant := - 2147483647;
  MAX_INT        : constant :=  2147483647;
  MAX_DIGITS     : constant := 15;
  MAX_MANTISSA   : constant := 31;
  FINE_DELTA     : constant := 2.0**(-30);
  TICK          : constant := 0.0001;
  -- Other System-Dependent Declarations
  subtype PRIORITY is INTEGER range 0 .. 0;
end SYSTEM;
```

### 4. Restrictions on Representation Clauses

This version of the Siemens BS2000 Ada Compiler does not support any representation clauses.

### 5. Conventions for Implementation-generated Names Denoting Implementation-dependent Components in Record Representation Clauses

Implementation-dependent components may be added to record objects by the compiler. These components remain

inaccessible for the user, i.e. they cannot be accessed via implementation-generated names, in particular they cannot be referred to in record representation clauses.

## 6. Interpretation of Expressions Appearing in Address Clauses

The present version of the Siemens BS2000 Ada Compiler does not support address clauses.

## 7. Restrictions on Unchecked Type Conversions

The Siemens BS2000 Ada Compiler supports the generic function `UNCHECKED_CONVERSION` with the following restriction: The actual generic subtype corresponding to the formal generic type `TARGET` must not be an unconstrained array type, and it must not be an unconstrained type with discriminants that have no defaults. The instances gained from `UNCHECKED_CONVERSION` return a target value whose bit pattern is a left-aligned copy of that of the source value. The numbers of bits transferred corresponds to the size of the target subtype. If the size of the source value is greater than the size of the target subtype, then the source value information is truncated on the right hand side, i.e. the low order bits are ignored. If the size of the source value is not greater than the size of the target subtype, then - again - as many bits are transferred as corresponds to the size of the target subtype, and no padding with zeroes, spaces or other characters is performed.

## 8. Implementation-Dependent Characteristics of Input-Output Packages

### 8.1 Conventions for NAME and FORM

External files are supported by the `SAM`, `ISAM`, `PAM`, `EAM`, `SYSDTA`, `SYSOUT` and `SYSLST` BS2000 files where the value of the parameter `FORM` determines which access method is selected.

The set of allowable values of `FORM` is given below together with the type of the BS2000 file corresponding to it. Leading blanks and lower case letters are not allowed in the `FORM` string.

value of FORM	BS2000 access method
"SAM"	Sequential Access Method
"ISAM"	Indexed Sequential Access Method
"PAM"	User Primary Access Method

"EAM"	Evanescent Access Method
"OMF"	The EAM file referred to by the BS2000 JCL as the * file
"RDTA"	The file (or terminal) associated with SYSDTA
"SOUT"	The file (or terminal) associated with SYSOUT
"SLST"	The file (or printer) associated with SYSLST

Each input-output package operates upon a sub-set of the allowable forms.

SAM, ISAM and PAM files are identified by the value of the parameter NAME of the CREATE and OPEN procedures whose characters must conform to the BS2000 file naming conventions as described below. The value of the parameter NAME is ignored for other values of FORM. Note that the ASCII characters in NAME are converted to EBCDIC within the packages for the comparisons with BS2000 file and link names.

The syntax associated with the string NAME is as follows:

```
NAME-syntax      ::= @ link_name | file_name
file_name        ::= $ user_id . name ( . name ) |
                   $ admin_name |
                   name ( . name )
link_name         ::= name_character ( name_character )
user_id           ::= name_character ( name_character )
admin_name        ::= name_character ( name_character )
name              ::= name_character ( name_character )
name_character    ::= upper_case_letter | digit |
                   special_character
special_character ::= $ @ # -
```

BS2000 imposes the following additional restrictions upon the syntax of a NAME string.

1. The maximum length of a link\_name or a user\_id is eight characters.
2. The maximum length of a file\_name starting with \$ user\_id is 54 characters.
3. The maximum length of an admin\_name is 47 characters unless it contains one or more periods in which case the maximum length is 53 characters.
4. The maximum length of a file\_name starting with name is 44 characters.

5. The first character of a name must not be a special character and the last character must not be a hyphen.
6. A file\_name must include at least one letter.

## 8.2 File management

This section describes the implementation restrictions which apply to the sequential, direct and text input-output packages equally.

The maximum number of objects which may be stored in an external file is dependent upon the maximum number of records or the maximum number of blocks which may be stored in its underlying BS2000 file. The values given below state this maximum for each FORM provided that the limits imposed by the system configuration are not otherwise reached. For the direct and sequential input-output packages, each object is stored in a separate record or block; for the text input-output package, each line is stored in a separate record.

FORM	Maximum Number of Records / Blocks
SAM	configuration dependent limit
ISAM	99 999 999 records
PAM	16 777 215 blocks
EAM	32 767 blocks
OMF	32 767 blocks
RDTA	configuration dependent limit
SOUT	configuration dependent limit
SLST	configuration dependent limit

Two alternative record formats are available for ISAM and SAM files, varying and constant length. TEXT IO always uses varying length records whereas DIRECT IO and SEQUENTIAL IO support both formats. A varying length record format is used if an instance of direct or sequential packages uses unconstrained element types. Otherwise a fixed length record format is used where the length equals the value of  $(\text{ELEMENT\_TYPE\_SIZE} + 7) / 8$  (that are the number of bytes needed for this special type).

The maximum size of the objects which can be stored in an external file is restricted. The universal integer value which results from the application of the SIZE attribute to every object accessed by the package must lie within a range which is dependent upon the FORM and whether constant or varying size records are being used. The exception USE\_ERROR is raised if this constraint is broken.

FORM	record form	OBJECT'SIZE (bits)
SAM	constant	1 .. 16_384
SAM	varying	1 .. 16_352
ISAM	constant	1 .. 16_320
ISAM	varying	1 .. 16_288
PAM	any object	1 .. 16_384
EAM	any object	1 .. 16_384
OMF	any object	1 .. 16_384
RDTA	varying	1 .. 2_032
SOUT	varying	1 .. 2_032
SLST	varying	1 .. 2_032

SAM, ISAM and PAM files are permanent files in that their lifetimes are independent of the BS2000 tasks in which they were created. Permanent files may be closed in one BS2000 task and opened subsequently in the same or another task without loss of their contents (for MODE = IN\_FILE or INOUT\_FILE).

When a SAM, ISAM or PAM file (selected by the value of NAME) is opened, there is no check that the form of the BS2000 file corresponds to the value of the FORM parameter of the OPEN procedure. There is also no check that the io-package opening a SAM, ISAM, PAM, EAM, or OMF file is the same package as the one which created the file. If either of these conditions is not valid, the program may deliver unexpected results.

EAM and OMF files may be closed and reopened within the same BS2000 program without loss of their contents (for MODE = IN\_FILE or INOUT\_FILE). Only one OMF file may be created or opened in each BS2000 task.

The RDTA, SOUT and SLST files are unique within a BS2000 task. They can neither be created nor deleted.

No assumptions should be made about the way objects are stored in the various BS2000 files except as described for the TEXT\_IO package. For example, the mapping of indices onto ISAM keys may differ between different versions of the input-output packages. Several internal files can be associated with the same external file for reading only, i.e. if all the internal files are opened with mode IN\_FILE.

### 8.3 SEQUENTIAL\_IO

The value of the FORM parameter of the CREATE and OPEN procedures is restricted to SAM, ISAM, PAM, EAM and OMF, with SAM being the default value.

The package SEQUENTIAL\_IO cannot be instantiated with

unconstrained array types and record types with discriminants without defaults.

#### 8.4 DIRECT\_IO

The value of the FORM parameter of the CREATE and OPEN procedures is restricted to ISAM, PAM, EAM and OMF, with ISAM being the default value.

The declaration of COUNT reads:  
type COUNT is range 0 .. INTEGER'LAST ;  
Therefore, the value of an index may be set in the range  
1 .. INTEGER'LAST.

The package DIRECT\_IO cannot be instantiated with unconstrained array types and record types with discriminants without defaults.

#### 8.5 TEXT\_IO

The value of the FORM parameter of the CREATE procedure is restricted to SAM and ISAM while the OPEN procedure may use SLST additionally. The default FORM parameter is SAM.

The MODE of the SLST file is restricted to OUT\_FILE.

The standard input file has the form RDTA and the standard output file has the form SOUT.

The ASCII characters passed across the interface to/from the TEXT\_IO package are converted to/from EBCDIC characters within the TEXT\_IO package; BS2000 files created and updated by the TEXT\_IO package contain EBCDIC characters.

The declaration of COUNT reads:  
type COUNT is range 0 .. INTEGER'LAST ;

The lines contained in text files are variable in length in the range 1..500 characters: The declaration of the subtype FIELD is given by:  
subtype FIELD is INTEGER range 0 ..500 ;

Lines are stored in the 2nd to 501st character of a BS2000 variable length file record. The first character of the record is a printer control character where ' ' means line-feed and 'A' page-feed. Thus BS2000 files created by calls to TEXT\_IO can be printed using the system command procedure DO.PRINT or displayed using the EDOR and EDT text file editors. The printer control characters are used to implement the line and page terminators and can

be manipulated using the standard line and page control procedures.

The input subprograms which either make or do not make a test for a page or file terminator are listed below:

Test for a page or file  
terminator  
END\_OF\_FILE  
END\_OF\_PAGE  
SET\_LINE  
SKIP\_PAGE

No test for a page or  
file terminator  
END\_OF\_LINE  
GET\_CHAR  
GET\_INT  
GET\_LINE  
GET\_STRING  
SET\_COL

### 8.6 Low Level Input-Output

Low Level Input-Output as described in chapter 14.6 has not been implemented.

### 9. Definition of a Main Subprogram

A library unit can be used as a main subprogram only if it is a non-generic procedure which has no formal parameters.

Name_and_Meaning_____	Value_____
\$BIG_ID1 Identifier of the size of the maximum input line length with varying last character.	239*'A' & '1'
\$BIG_ID2 Identifier of the size of the maximum input line length with varying last character.	239*'A' & '2'
\$BIG_ID3 Identifier of the size of the maximum input line length with varying middle character.	120*'A' & '3' & 119*'A'
\$BIG_ID4 Identifier of the size of the maximum input line length with varying middle character.	120*'A' & '4' & 119*'A'
\$BIG_INT_LIT An integer literal of value 298 with enough leading zeroes so that it is the size of the maximum line length.	237*'0' & "298"
\$BIG_REAL_LIT A real literal that can be either of floating- or fixed-point type, has value 690.0, and has enough leading zeroes to be the size of the maximum line length.	234*'0' & "69.0E1"
\$BLANKS A sequence of blanks twenty characters fewer than the size of the maximum line length.	220*' '

SCOUNT_LAST	2_147_483_647
A universal integer literal whose value is TEXT_IO.COUNT'LAST.	
SEXTENDED_ASCII_CHARS	"" "abcdefghijklmnopqrstuvwxyz" &
A string literal containing all the ASCII characters with printable graphics that are not in the basic 55 Ada character set.	
	ascii.exclam & ascii.dollar & ascii.percent & ascii.query & ascii.at_sign & ascii.r_bracket & ascii.back_slash & ascii.l_bracket & ascii.circumflex & ascii.grave & ascii.l_brace & ascii.r_brace & ascii.tilde & ' "'
\$FIELD_LAST	500
A universal integer literal whose value is TEXT_IO.FIELD'LAST.	
\$FILE_NAME_WITH_BAD_CHARS	BAD_FILE_NAME
An illegal external file name that either contains invalid characters, or is too long if no invalid characters exist.	
\$FILE_NAME_WITH_WILD_CARD_CHAR	WILD*FILE*NAME
An external file name that either contains a wild card character, or is too long if no wild card character exists.	
\$GREATER_THAN_DURATION	131_071.5
A universal real value that lies between DURATION'BASE'LAST and DURATION'LAST if any, otherwise any value in the range of DURATION.	
\$GREATER_THAN_DURATION_BASE_LAST	200_000.0
The universal real value that is greater than DURATION'BASE'LAST, if such a value exists.	
\$ILLEGAL_EXTERNAL_FILE_NAME1	BAD_FILE_NAME
An illegal external file name.	
\$ILLEGAL_EXTERNAL_FILE_NAME2	MUCH-TOO-LONG-FOR-A-CORRECT- BS2000-FILE-NAME
An illegal external file name that is different from \$ILLEGAL_EXTERNAL_FILE_NAME1.	

\$INTEGER_FIRST	-2147_483_647
The universal integer literal expression whose value is INTEGER'FIRST.	
\$INTEGER_LAST	2_147_483_647
The universal integer literal expression whose value is INTEGER'LAST.	
\$LESS_THAN_DURATION	-131_071.5
A universal real value that lies between DURATION'BASE'FIRST and DURATION'FIRST if any, otherwise any value in the range of DURATION.	
\$LESS_THAN_DURATION_BASE_FIRST	-200_000.0
The universal real value that is less than DURATION'BASE'FIRST, if such a value exists.	
\$MAX_DIGITS	15
The universal integer literal whose value is the maximum digits supported for floating-point types.	
\$MAX_IN_LEN	240
The universal integer literal whose value is the maximum input line length permitted by the implementation.	
\$MAX_INT	2_147_483_647
The universal integer literal whose value is SYSTEM.MAX_INT.	
\$NAME	long_long_integer
A name of a predefined numeric type other than FLOAT, INTEGER, SHORT FLOAT, SHORT INTEGER, LONG FLOAT, or LONG INTEGER if one exists, otherwise any undefined name.	
\$NEG_BASED_INT	8#37777777776#
A based integer literal whose highest order nonzero bit falls in the sign bit position of the representation for SYSTEM.MAX_INT.	

SNON\_ASCII\_CHAR\_TYPE

(non\_null)

An enumerated type definition  
for a character type whose  
literals are the identifier  
NON\_NULL and all non-ASCII  
characters with printable  
graphics.

## APPENDIX D

### WITHDRAWN TESTS

Some tests are withdrawn from the ACVC because they do not conform to the Ada Standard. The following 19 tests had been withdrawn at the time of validation testing for the reasons indicated. A reference of the form "AI-ddddd" is to an Ada Commentary.

- . C32114A: An unterminated string literal occurs at line 62.
- . B33203C: The reserved word "IS" is misspelled at line 45.
- . C34018A: The call of function G at line 114 is ambiguous in the presence of implicit conversions.
- . C35904A: The elaboration of subtype declarations SFX3 and SFX4 may raise `NUMERIC_ERROR` instead of `CONSTRAINT_ERROR` as expected in the test.
- . B37401A: The object declarations at lines 126 through 135 follow subprogram bodies declared in the same declarative part.
- . C41404A: The values of 'LAST and 'LENGTH are incorrect in the if statements from line 74 to the end of the test.
- . B45116A: `ARRPRIBL1` and `ARRPRIBL2` are initialized with a value of the wrong type -- `PRIBOOL_TYPE` instead of `ARRPRIBOOL_TYPE` -- at line 41.
- . C48008A: The assumption that evaluation of default initial values occurs when an exception is raised by an allocator is incorrect according to AI-00397.
- . B49006A: Object declarations at lines 41 and 50 are terminated incorrectly with colons, and end case; is missing from line 42.
- . B4A010C: The object declaration in line 18 follows a subprogram body of the same declarative part.
- . B74101B: The begin at line 9 causes a declarative part to be treated as a sequence of statements.
- . C87B50A: The call of `"/=` at line 31 requires a use clause for package A.

- . C92005A: The "/"= for type PACK.BIG\_INT at line 40 is not visible without a use clause for the package PACK.
- . C940ACA: The assumption that allocated task TT1 will run prior to the main program, and thus assign SPYNUMB the value checked for by the main program, is erroneous.
- . CA3005A..D (4 tests): No valid elaboration order exists for these tests.
- . BC3204C: The body of BC3204C0 is missing.

END

DATE

FILMED

8-88

DTIC